Product Perceptual Mapping on Fashion Designs with Gaussian Mixture Variational Autoencoder and Triplet Loss

by

Mike Wang

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2018

Author .....
Department of Electrical Engineering and Computer Science
August 21, 2018

Certified by .....
Professor Drazen Prelec
Professor of Management Science and Economics
Thesis Supervisor

Certified by .....
Dr. Alex Burnap
Postdoctoral Fellow
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Product Perceptual Mapping on Fashion Designs with Gaussian Mixture Variational Autoencoder and Triplet Loss

by

Mike Wang

Submitted to the Department of Electrical Engineering and Computer Science
on August 21, 2018, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Product perceptual maps are visualizations of the perceptions of products by customers. They provide many advantages to businesses, such as identifying gaps in the market, understanding competition, and finding how new products fit into a market. Conventional product perceptual mapping methods exhibit limitations, particularly in capturing the highly nonlinear structure in product perceptual categories. Therefore, given only a set of images and triplet data representing product co-occurence by consumers, we propose and use a Gaussian mixture variational autoencoder (GMVAE) with triplet loss to create product embeddings. These product embeddings are then flattened into a 2D perceptual map able to be interpreted by human judgment. We test the GMVAE approach on three datasets: (1) a dataset of simple generated data; (2) the MNIST dataset, a dataset of handwritten digits; and (3) the Amazon Fashion dataset, a dataset of product images, product categories, and similar products. The GMVAE method is quantitatively evaluated on its ability to capture product "latent" categories, and qualitatively evaluated on the quality of its 2D perceptual maps compared with those produced by using a conventional perceptual mapping method. We find that across the experiments, the GMVAE method could reasonable capture "latent" perceptual product categories and is more effective than the conventional perceptual mapping baseline in correctly identifying and predicting latent product categories.

Thesis Supervisor: Professor Drazen Prelec
Title: Professor of Management Science and Economics

Thesis Supervisor: Dr. Alex Burnap
Title: Postdoctoral Fellow

Acknowledgments

Drazen Prelec, for being a great supervisor for this project.

Alex Burnap, for being both a great supervisor and a terrific mentor.

My family, especially my mother, for supporting me throughout my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Understanding how products are perceived by customers is a fundamental task in market and product design. There are many tasks for the firm, including: (1) to identify gaps in the market, for example, determining a combination of perceptual aspects which is unlike any other combination currently in the market [31]; (2) understanding competition, for example, seeking to merge with other businesses that produces similarly perceived goods [1]; and (3) understanding how new products fit into the market, for example, whether a new product would compete with other goods produced by the same company [28]. In fact, product perception is still a heavily researched area [24].

The primary market research and product design approach to visualize this information is through perceptual mapping [15], which displays the perceptions of products by consumers. This general approach to visualization of customer perceptions includes: collecting data directly from customers using surveys or focus groups on the perceptions about products (qualitative approach) [26], or using market data capturing what consumers bought in order to infer how they perceive products (quantitative approach) [14]. Focusing on this second direction, there are several quantitative methods used in order to create perceptual maps. In general, these methods create lower dimensional "embeddings" that captures customer perceptions over products. Some of the more popular quantitative methods includes using principal component analysis (PCA) [15], autoencoders, and multi-dimensional scaling (MDS) [21].

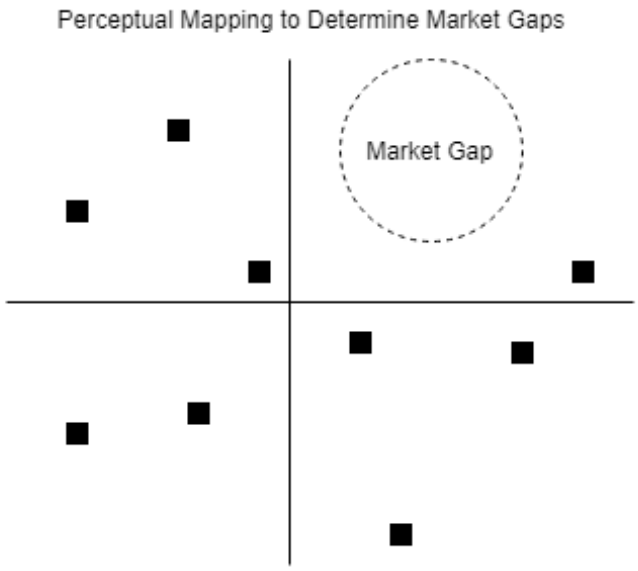


Figure 1-1: Example of perceptual mapping application, identifying a market gap

However, there are several problems with current quantitative methods for perceptual mapping. For example, a primary problem with using PCA is that the embedding transformation performed with PCA is linear [19], and so is unable to capture a lot of the complexity of how a consumer may perceive a product. A primary problem with using an autoencoder to create meaningful embeddings is that the embedding space is not constrained to fit any particular probabilistic or semantic form [29]. Therefore, distance in the embedding space may not have semantic meaning, and it may be infeasible to use distance to determine similarity between products. A primary problem with using MDS is that it does not explicitly capture latent variables about products which may be important for identifying latent product categories, only capturing variables that are explicitly accounted for when calculating distances between products [5].

Autoencoders, PCA, and MDS could all be described as unsupervised machine learning methods [29]. This suggests the possibility of leveraging newer techniques for unsupervised machine learning that could address some of the problems with the conventional methods, and ultimately extend conventional perceptual mapping approaches.

The motivation of this project is to therefore develop an unsupervised method

to create perceptual maps with unknown *a priori* variables that capture semantic meaning useful to the marketer or product designer. Our specific proposed method is a Gaussian mixture variational autoencoder (GMVAE) with triplet loss. This method is similar to the autoencoder in that it has two component neural nets, an encoder neural net and a decoder neural net [22]. However, the GMVAE additionally constrains the embedding space of the neural net to have a Gaussian mixture prior. This causes similar products to map to the same Gaussian in the embedding space and therefore be closer together in the embedding space. More importantly, this allows the capture of latent variables that are predicted to have semantic meaning.

Our proposed method leverages: (1) visual data of products; and (2) data of products customers often view together in order to shape the perceptual mapping of the products. Research has shown that visual data is important to to marketing [20] [8]; this data is incorporated into our model directly as input vectors. Also, data of products customers view together could provide clues on product perception; this data is incorporated into our model through "triplet loss", which is a supervised loss signal that tells us that anchor product a and "positive" product p should be more similar to each other than anchor product a and "negative" product n [17].

We conduct three separate experiments on different datasets to test the proposed GMVAE method. The first experiment is on simulated data, which was generated to have distinct embedding clusters for the GMVAE to find. This experiment is used as a simple check to see if the model works as expected. The second experiment is on the well-known MNIST dataset [6], a collection of images of handwritten digits. This experiment is used to see how the model works on a real, but simple, dataset. The last experiment is on Amazon Fashion data [16], a collection of images of different products, the category of the products, and a list of products that are often viewed together. This last experiment is used to determine how effectively the GMVAE could be used to create perceptual groupings on real, complex product data.

First, to quantitatively evaluate the experiments, we measure how well the method captures "unknown" latent variables, or variables that are not known to the model, but known to us as experiments as important to capturing latent product perceptual

categories. Second, we qualitatively compare the perceptual map created by our approach, 2D visualizations of the product perceptual embeddings from the GMVAE model, to the perceptual map created by the baseline conventional perceptual map, MDS.

Across the experiments, we find quantitatively that the GMVAE method with triplet loss is able to capture latent categories, and qualitatively that it creates better perceptual maps than the MDS baseline. We next discuss the effects of changing the number of training clusters, the effectiveness of triplet loss, and limitations of the method.

The rest of this thesis is structured as follows: Section 2 is related work, Section 3 is formal problem statement and model introduction, Section 4 describes experiments and results, Section 5 is discussions and limitations, and Section 6 concludes this work.

Chapter 2

Related Work

In this chapter, we discuss current unsupervised quantitative approaches to produce perceptual maps used in market research. These approaches include principal component analysis (PCA), autoencoders, multi-dimensional scaling (MDS), and variational autoencoders.

2.1 PCA

Principal component analysis (PCA) is a common method of dimensionality reduction, e.g. it has been used by Hauser [15], Nestrud [30], Faye [10] for dimensionality reduction purposes in the specific application of perceptual mapping.

PCA is method where a vector is linearly transformed to a different set of orthogonal unit components that captures most of the variance in the original data [19].

Formally, let us consider a dataset with n samples and f features, forming a data matrix m with dimensions $n \times f$. Let m_i denote the i^{th} row of the data matrix. The first principal component, $w = (w_0, w_1, ..., w_{f-1})$, is a size f unit vector that maximizes the variance of $mw = \sum_i m_i w$. Note that if we normalize m to have a column-wise 0 empirical mean, this variance becomes equal to $\text{Var}[mw] = \sum_i (m_i w)^2 = w^T m^t m w$. Since w is a unit vector, this is equal to $\frac{w^T m^t m w}{w^T w}$. A standard result is that this is maximized when w is the eigenvector of the largest

eigenvalue of the matrix [2].

Each subsequent principal component is found by performing the same calculations on m minus all of the earlier principal components.

The embedding from PCA can be better than the standard autoencoder, since distance has more semantic meaning. In general, the closer the embeddings are for PCA, the closer the original vectors are. However, transformations are linear and cannot capture some complex relationships between the input vector and the embedding.

2.2 Autoencoder

Autoencoders is a common method of dimensionality reduction, e.g. it has been used by Hinton [18], Sakurada [32], Wang [33] for dimensionality reduction purposes.

The autoencoder operates on data of the format $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ of N samples of some continuous random vector $\mathbf{x}$ [12]. A standard autoencoder has two component neural nets - an encoder neural net and a decoder neural net. These neural nets are trained in a way to minimize the "reconstruction loss" of a product, or the difference between the original object and the object after being encoded and decoded.

Formally, we can write the encoder net as $\phi : \mathbf{x} \to \mathbf{y}$, which transforms input $\mathbf{x}$ to latent embedding $\mathbf{y}$. We can write the decoder net as $\psi : \mathbf{y} \to \mathbf{x}$, which transforms a latent embedding $\mathbf{y}$ to a reconstructed $\mathbf{x}$. The loss minimized would often be in the form, $||\mathbf{x} - (\psi \circ \phi)\mathbf{x}||^2$.

2.3 MDS

Multidimensional scaling (MDS) is a common method of dimensionality reduction, e.g. it has been used by Birch [3], Green [13], Macfie [25] for dimensionality reduction purposes in the specific application of market research. Specifically, Markiczy [27] used MDS to create casual maps Desarbol [7] used MDS to create joint consumer product maps.

MDS is a nonlinear method similar to PCA. MDS reduces the dimensionality of

objects while preserving distance as well as possible. Specifically, we use metric-MDS [4], which minimizes a cost function called stress. Letting $d_{i,j}$ denotes the distance between input vectors x_i and x_j, stress is defined as:

$$Stress_D(x_1, x_2, ..., x_n) = (\frac{\sum_{i,j}(d_{i,j} - ||x_i - x_j||)^2}{\sum_{i,j} d_{i,j}^2})^{\frac{1}{2}} \tag{2.1}$$

Similarly to PCA, the method will tend to place objects more similar to each other closer together within the embedding space. The problem with MDS is that the distance function needs to be explicit and so could only capture explicit variables.

2.4 Variational Autoencoders

A new method that we propose to apply to product perceptual mapping is variational autoencoders.

Variational autoencoders (VAE) were introduced by Kingma [22]. In the original unsupervised VAE paper, the model operates on data of the format $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ of N samples of some continuous random vector $\mathbf{x}$. The data is generated by some random process involving unobserved latent variable $\mathbf{z}$ involving two steps. In the first step, value $\mathbf{z}^i$ is sampled from some prior distribution $p_\theta(\mathbf{z})$. In the second step, value $\mathbf{x}^i$ is sampled from the conditional distribution $p_\theta(\mathbf{x}|\mathbf{z})$.

In order to maximize the probability of the observations $\mathbf{X}$, we maximize $\log p_\theta(\mathbf{x}) = \sum_{i=1}^N \log p_\theta(\mathbf{x}^{(i)})$, where:

$$\log p_\theta(\mathbf{x}^{(i)}) = \sum_z p_\theta(\mathbf{z}|\mathbf{x}^{(i)}) \log p_\theta(\mathbf{x}^{(i)}) \tag{2.2}$$

However, the model distribution $p_\theta(\mathbf{z}|\mathbf{x}^{(i)})$ is intractable (it is some unknown and potentially extremely complex distribution). Therefore, instead, we approximate the model conditional distribution with a fixed-formed distribution $q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$ with parameter ϕ. In the model, the distribution $q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$ is a multivariate Gaussian.

Therefore, we write:

$$\log p_\theta(\mathbf{x}^{(i)}) = \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) \log p_\theta(\mathbf{x}^{(i)})$$

$$= -\sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) \log p_\theta(\mathbf{z}|\mathbf{x}^{(i)}) + \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) \log p_\theta(\mathbf{x}^{(i)}, \mathbf{z})$$

$$= KL(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\theta, \phi; \mathbf{x}^{(i)}) \tag{2.3}$$

The first term is the KL divergence of the approximate from the true model posterior. This term is always non-negative, so the second term, $\mathcal{L}(\theta, \phi; x^{(i)})$, is known as the variational lower bound. It can be written as:

$$\log p_\theta(\mathbf{x}^{(i)}) \geq \mathcal{L}(\theta, \phi; \mathbf{x}^{(i)})$$

$$= -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}(\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})) \tag{2.4}$$

This becomes the loss function that variational autoencoders try to minimize.

Chapter 3

Problem Formulation

Data Formulation

1. Feature Data:
   (a) $\mathbf{x}$: The input vector. This is the image data processed by the GMVAE.
   (b) $\mathbf{y}$: The latent cluster x belongs to. This is not known during training, and is used to evaluate the model.
2. Triplet Data, of form $(\mathbf{a}, \mathbf{p}, \mathbf{n})$:
   (a) $\mathbf{a}$: The anchor vector of the triplet. This is the vector with which similarity is measured with and compared to.
   (b) $\mathbf{p}$: The positive vector of the triplet. The positive vector is closely related to the anchor vector.
   (c) $\mathbf{n}$: The negative vector of the triplet. This negative vector is not related to the anchor vector.

Figure 3-1: Summary of data and notation

We now formally describe the task of product perceptual mapping. The data is a set of product images in the form $(\mathbf{X}) = \{(\mathbf{x}^{(1)}_{\mathbf{y}_{k_1}}), ..., (\mathbf{x}^{(N)}_{\mathbf{y}_{k_N}})\}$, where $\mathbf{x}^{(i)}_{\mathbf{y}_{k_i}}$ represents image data for the i^{th} product belonging to the "unknown" latent cluster $\mathbf{y}_{k_i}$. In addition, we are given a set of triplet data over these products, $(\mathbf{T}) = \{(\mathbf{a}^{(1)}, \mathbf{p}^{(1)}, \mathbf{n}^{(1)}), ..., (\mathbf{a}^{(N)}, \mathbf{p}^{(N)}, \mathbf{n}^{(N)})\}$, where $\mathbf{a}^{(i)}, \mathbf{p}^{(i)}, \mathbf{n}^{(i)} \in \mathbf{X}$ and anchor image $\mathbf{a}^{(i)}$ and "positive" image $\mathbf{p}^{(i)}$ are more close related to each other than anchor image $\mathbf{a}^{(i)}$

is related to "negative" image $\mathbf{n}^{(i)}$.

3.1 Proposed Approach: Gaussian Mixture Variational Autoencoder

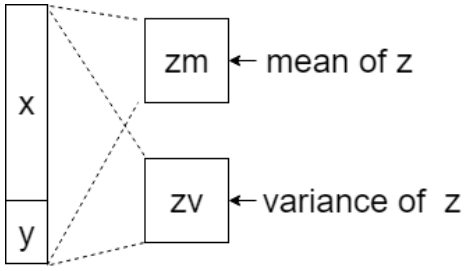
Our proposed approach to the problem is a Gaussian mixture variational autoencoder (GMVAE). This method extends the variational autoencoder described in Section 2.4 by using a mixture of Gaussians in the embedding space, allowing latent product categories $\mathbf{y}_{k_1}, \mathbf{y}_{k_2}, ..., \mathbf{y}_{k_N}$ to be predicted. The mathematical justification for the GMVAE is the same as the variational autoencoder described in Section 2.4.

Again, using the formulation from Section 2.4, maximizing the probability of observations $\mathbf{X}$ involves maximizing $\log p_\theta(\mathbf{x}^{(i)}) = \sum_z p_\theta(\mathbf{z}|\mathbf{x}^{(i)}) \log p_\theta(\mathbf{x}^{(i)})$, as shown in Equation 2.2. However, the model distribution $p_\theta(\mathbf{z}|\mathbf{x}^{(i)})$ is intractable, and so instead we approximate the model conditional distribution with a Gaussian mixture distribution $q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$. Using the approximate conditional distribution in place of the model conditional distribution, we find $\log p_\theta(\mathbf{x}^{(i)}) = KL(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\theta, \phi; \mathbf{x}^{(i)})$, as shown in Equation 2.3. Therefore, the model relies on minimizing the variational lower bound, $\mathcal{L}(\theta, \phi; x^{(i)}) = -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}(\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z}))$, Equation 2.4, where we constrain the approximate distribution $q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$ to be a Gaussian mixture.

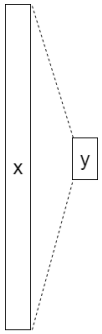
This variational lower bound is the loss function for our Gaussian mixture variational autoencoder. The first part of the variational lower bound, $-D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z}))$, we call the KL loss. The second part of the variational lower bound, $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}(\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z}))$, we call the reconstruction loss.

For the Gaussian mixture variational autoencoder, we also condition on latent variable $\mathbf{y}$, which represents which cluster we expect to be the product latent category. The variational lower bound becomes $\mathcal{L}(\theta, \phi; x^{(i)}) = -D_{KL}(q_\phi(\mathbf{z}, \mathbf{y}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z}, \mathbf{y})) + \mathbb{E}_{q_\phi(\mathbf{z},\mathbf{y}|\mathbf{x}^{(i)})}(\log p_\theta(\mathbf{x}^{(i)}|\mathbf{y}, \mathbf{z}))$. Note we could separate $q_\phi(\mathbf{z}, \mathbf{y}|\mathbf{x}^{(i)}) = q_\phi(\mathbf{z}|\mathbf{x}^{(i)}, \mathbf{y}) q_\phi(\mathbf{y}|\mathbf{x}^{(i)})$, and $p_\theta(\mathbf{z}, \mathbf{y}) = p_\theta(\mathbf{z}|\mathbf{y}) p_\theta(\mathbf{y})$. However, if we set the prior $p_\theta(\mathbf{y})$ to be uniform, we

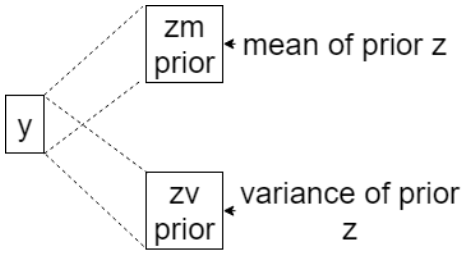
can ignore this constant factor. Therefore, we only need to calculate the factors $q_\phi(\mathbf{z}|\mathbf{x}^{(i)}, \mathbf{y})$, $q_\phi(\mathbf{y}|\mathbf{x}^{(i)})$, $p_\theta(\mathbf{z}|\mathbf{y})$, and $p_\theta(\mathbf{x}^{(i)}|\mathbf{y}, \mathbf{z}) \approx p_\theta(\mathbf{x}^{(i)}|\mathbf{z})$. This forms the basis of 4 different deep neural nets:



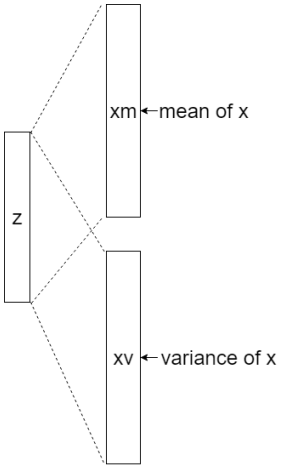
(a) Encoder for perceptual map embedding



(b) Encoder for latent product category



(c) Encoder for prior perceptual map embedding



(d) Decoder for product

Figure 3-2: Component neural networks of GMVAE; the four component neural networks combine to form the GMVAE model

1. A net that finds $q_\phi(\mathbf{z}|\mathbf{x}^{(i)}, y)$, constraining this to be a Gaussian

2. A net that finds $q_\phi(\mathbf{y}|\mathbf{x}^{(i)})$.

3. A net that finds the prior probability $p_\theta(\mathbf{z}|\mathbf{y})$, constraining this to be a Gaussian. Note that applying this constraint makes the z prior a Gaussian mixture.

4. A net that finds $p_\theta(\mathbf{x}^{(i)}|\mathbf{z})$, constraining this to be a Gaussian.

3.2 Triplet Loss

In addition to the GMVAE, we leverage triplet data in order to give the models an additional supervised signal. Specifically, we are given triplets vectors $(\mathbf{a}, \mathbf{p}, \mathbf{n})$ where anchor vector $\mathbf{a}$ is more similar to "positive vector" $\mathbf{p}$ than anchor vector $\mathbf{a}$ is to "negative" vector $\mathbf{n}$ [17]. By adding triplet loss to our loss function, we hope to incorporate perceptual information to cluster upon.

We use the distance between vector embeddings as the metric for similarity. This is consistent with the embeddings produced from the GMVAE, since similar vectors are found in the same cluster and are usually closer together than to vectors from other clusters.

The expression for triplet loss is:

$$\mathbb{E}_{q_\phi(\mathbf{z}^{(a)},\mathbf{y}^{(a)},\mathbf{z}^{(p)},\mathbf{y}^{(p)},\mathbf{z}^{(n)},\mathbf{y}^{(n)}|\mathbf{a},\mathbf{p},\mathbf{n}}} \max(0, \alpha + E(\mathbf{a}) \cdot E(\mathbf{n}) - E(\mathbf{a}) \cdot E(\mathbf{p}) \tag{3.1}$$

where E is the encoder as shown in figure 3-2 a. The α term in the expression is the margin, a hyperparameter that we have set to 0.2 in our experiments.

It should be noted that this evaluation is computationally expensive. The number of operations scales cubicly, $O(n^3)$, with how many clusters (and therefore how many $\mathbf{z}$-embeddings) used in the model.

3.3 Evaluation Metric

In all of the experiments, we have "unknown" labels for products with which to evaluate the accuracy of our model. These labels correspond to latent product categories, and not made known during model training, and are only used during evaluation. We evaluate the accuracy the following way for each experiment:

1. For each cluster (or distinct $\mathbf{y} \in 0, 1^k$ outputted by a component net) found by the GMVAE, we collect all of the member vectors and their labels.

2. We find the label most represented in the cluster. Each vector in the cluster is then assigned this label as the predicted label.

3. We determine the overall accuracy, as measured by the fraction of all predicted labels matching the real labels.

For example, consider a cluster whose latent produce category labels are "0", "2", "1", "2", "3", "2". Since "2" is the most common label in this cluster, the predicted label for all members of the cluster is "2". This cluster has a 0.5 accuracy, since 3 out of the 6 predicted labels matched the real labels.

Chapter 4

Experiments

We now describe the three experiments to test our proposed approach for perceptual mapping. The experiments are in increasing difficultly. Experiment 1 is on simulated data, Experiment 2 is on MNIST data, a well known dataset of handwritten digits, and Experiment 3 is on the Amazon Fashion dataset, a dataset of product images, produce categories, and similar products.

The first two experiments share the same model architecture, but the last experiment on Amazon Fashion data has a much different model architecture.

4.1 Experiment 1: Simulated Data

Our first experiment is on simulated data, where we assume 10 different classes or clusters representing unknown latent produce categories. We describe in this subsection the data, the model architecture, and the results where we compare the performance of the GMVAE with the MDS baseline.

4.1.1 Data

The data consists of vectors of size 784 and a label telling which cluster each vector belongs to. This label is not used for training the model.

The data is generated in the following way:

Number of Training Data	100000
Number of Test Data	10000
Number of "Unknown" Latent Classes	10
Data Dimensionality	Size 784 Vector
Data Type	Real-Valued

Table 4.1: Data summary of simulated product data

1. We first generate the Gaussian mixture embedding space, $\mathbf{z}$, which has dimension 50. We randomly create 10 different vectors of size 50 where each value is between 0 and 5. For the variance, we randomly create 10 different vectors of size 50 where each value is between 0 and 1.

2. We sample from the embedding space. We create 100000 training points and 10000 test points.

3. We create a polynomial mapping between each embedding $\mathbf{z}$ and the vectors $\mathbf{x_m}$ and $\mathbf{x_v}$, the mean and variance from which vector $\mathbf{x}$ is sampled from. We do so by creating a polynomial relationship of degree 5. We generate 5 different matrices of dimension $dim(\mathbf{x}) \times dim(\mathbf{z})$, $m_1, m_u 2, m_3, m_4, m_5$ for the mean, and five difference matrices v_1, v_2, v_3, v_4, v_5 for the variance. The relationship is $\mathbf{x_m} = m_1\mathbf{z} + m_2\mathbf{z}^2 + m_3\mathbf{z}^3 + m_4\mathbf{z}^4 + m_5\mathbf{z}^5$ and $\mathbf{x_v} = v_1\mathbf{z} + v_2\mathbf{z}^2 + v_3\mathbf{z}^3 + v_4\mathbf{z}^4 + v_5\mathbf{z}^5$. Each value of matrix m is a randomly chosen value between 2 and 4, and each value of matrix v is a randomly chosen value between 1 and 2. The dimension of $\mathbf{x}$ was chosen to be 784.

4. We finally create the corresponding $\mathbf{x}$ by sampling from the Gaussian described by generated $\mathbf{x_m}$ and $\mathbf{x_v}$.

Triplet data is also generated for this dataset. Each triplet of the data, $(\mathbf{a}^i, \mathbf{p}^i, \mathbf{n}^i)$ is generated by the following steps:

1. Randomly choose a class (one of the 10 classes of the generated data). In the class, randomly choose an image. This image is the anchor, $\mathbf{a}^i$.

2. Randomly choose another image form the same class. This image is the "positive" image, $\mathbf{p}^i$.

3. Randomly choose a different class. In the class, randomly choose an image. This image is the "negative" image, $\mathbf{n}^i$.

We preprocess all of the data by normalizing it: we subtract the mean of the training data and divide by the standard deviation.

4.1.2 Model Architecture

For the model, we give the number of mixtures, k, in the Gaussian mixture, as well as some other hyperparameters (e.g. loss lambdas, size of layers). The model has four different component nets:

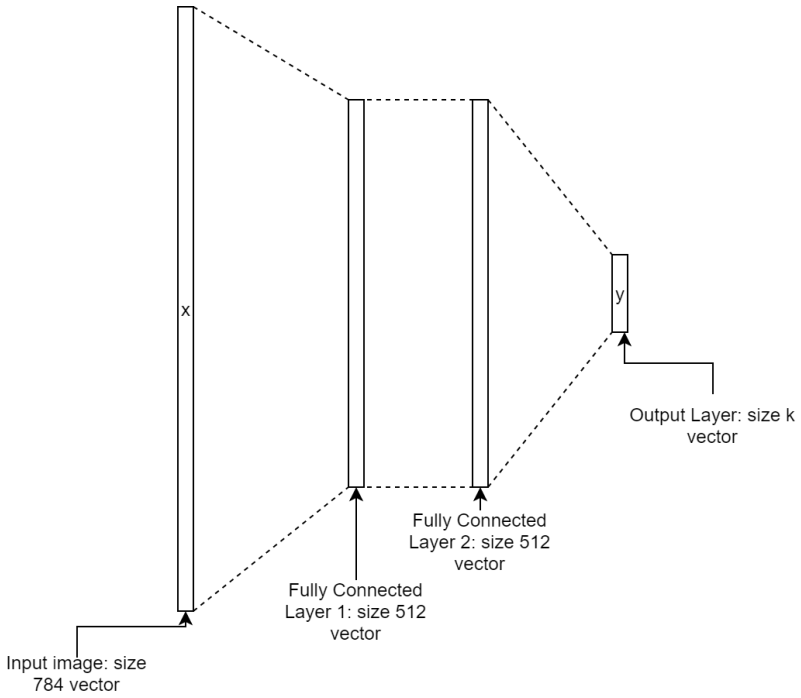


Figure 4-1: Architecture of the first component neural net, the encoder for latent product category

1. An encoder net that takes the image data, $\mathbf{x}$, and predicts a cluster $\mathbf{y}$. This net consists of three fully connected layers, where the first two layers uses ReLU

[23] activation. The last layer is of size k, where softmax [9] activation gives the probability of the image being in each cluster.

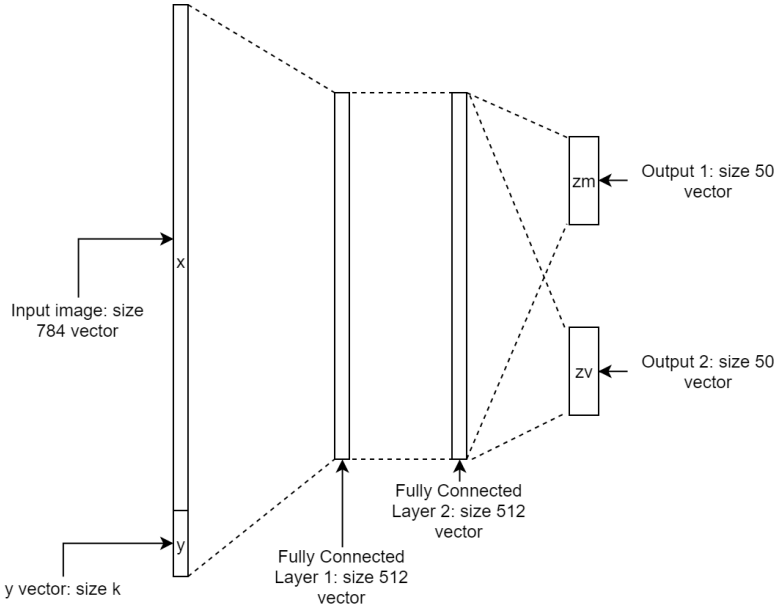


Figure 4-2: Architecture of the second component neural net, the encoder for perceptual map embedding

2. An encoder net that take the image data, $\mathbf{x}$, and predicted cluster $\mathbf{y}$, to create the embedding $\mathbf{z}$ (with dimension reduction). This net takes as input the concatenated vector of $\mathbf{x}$ with $\mathbf{y}$, and consists of four fully connected layers. The first two layers use ReLU activation, the third layer takes the second fully connected layer and outputs the mean of the embedding ($\mathbf{z_m}$), and the fourth layer takes the second fully connected layer and outputs the variance of the embedding ($\mathbf{z_v}$). The fourth layer uses softplus [11] activation to ensure that variance is positive.

3. A encoder net that takes the predicted cluster, $\mathbf{y}$, and predicts the Gaussian prior for the cluster in the embedding space (since there are k different values for $\mathbf{y}$, this enforces a Guassian mixture prior over the entire embedding space). There are 2 different fully connected layers. The first layer uses ReLU activation and outputs the prior mean, $\mathbf{z_m^p}$, and the second layer uses softplus activation and outputs the prior variance, $\mathbf{z_v^p}$.

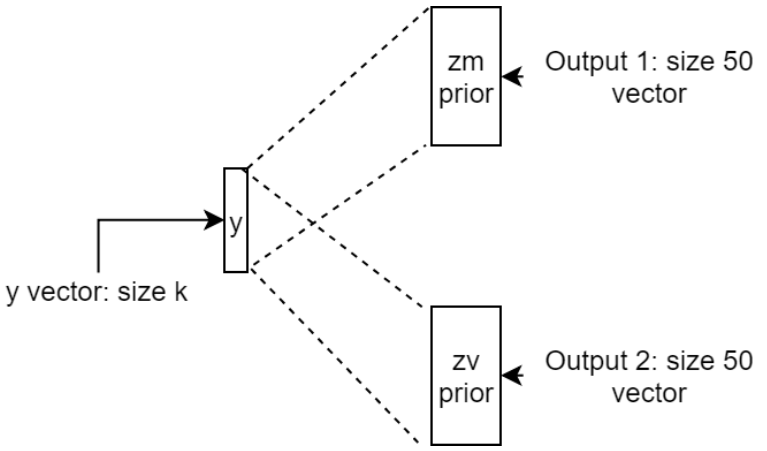


Figure 4-3: Architecture of the third component neural net, the encoder for prior cluster embedding

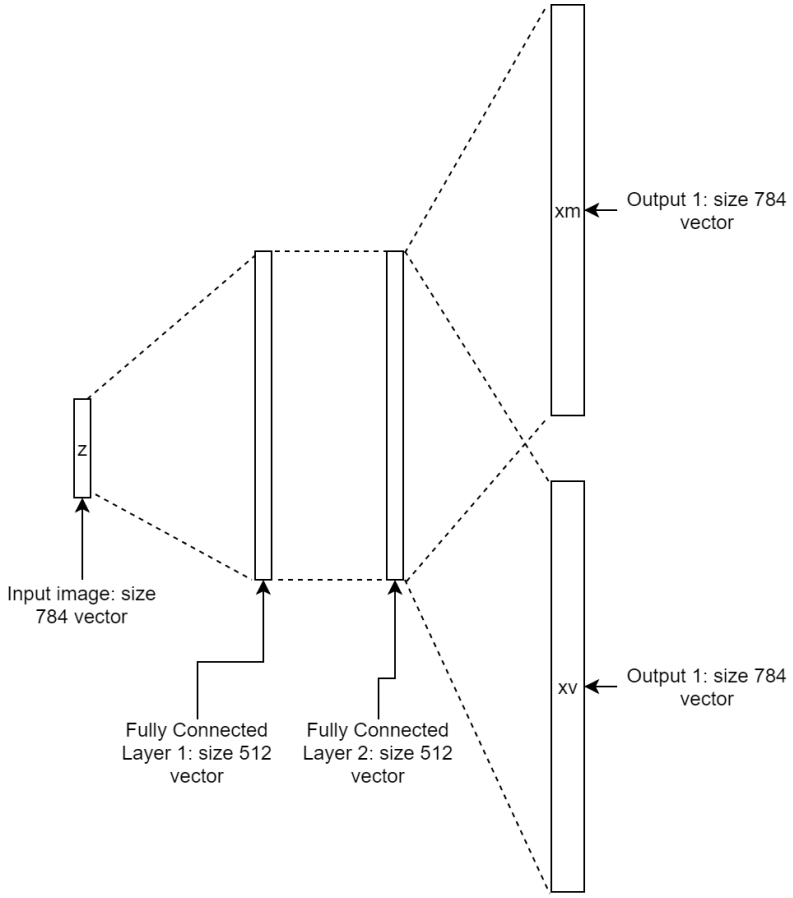


Figure 4-4: Architecture of the fourth component neural net, the decoder of the perceptual map embedding

4. A net that takes the embedding, $\mathbf{z}$, and tries to reconstruct the image $\mathbf{x_r}$. It does so by assuming a Gaussian distribution of $\mathbf{x}$ given $\mathbf{z}$. The net outputs

two different values, the mean of the predicted reconstruction $\mathbf{x_m}$, and the variance of the predicted reconstruction $\mathbf{x_v}$. There are 4 different fully connected layers. The first two layers use ReLU activation, one layer takes the second fully connected layer and outputs the mean of mean of the predicted reconstruction, and the last layer takes the second fully connected layer and outputs the variance of the mixture. The last layer uses softplus activation.

4.1.3 Results

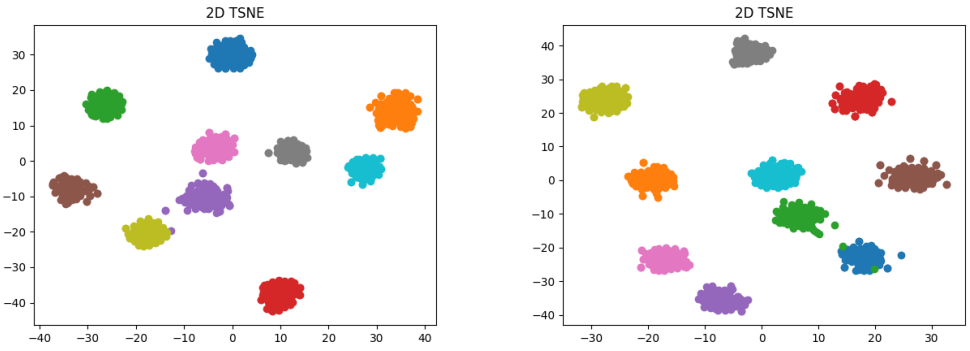
We vary the number of clusters and whether triplet loss is used when running the model. Note that results are specified for when them model was trained with 1600000 training points rather than a certain number of epochs. Since the model that uses triplet loss also incorporates the extra data from training with the triplets, the triplet loss model actually trained for fewer epochs.

Number of Clusters (k)	Triplet	Test Accuracy
15	No	0.905
	Yes	1.00

Table 4.2: Product latent category prediction accuracy of the GMVAE after 1600000 training points

We visualize the effectiveness of the clustering in 2D by applying t-distributed Stochastic Neighbor Embedding (t-SNE) to the embedding created by the GMVAE. The results are shown below; each different color represents one of the 10 different classes generated for the data. We see that the GMVAE performs extraordinarily well, nearly separating the data perfectly.

Interestingly, even though the model that did not include triplet loss has a 0.905 accuracy for cluster membership, the embeddings are perfectly separated. This implies that the model did not learn to use cluster membership, y, when creating the embedding z. One of the classes for the digits is degenerate and since none of the k clusters captures it by itself, it must have been incorporated into one of the other k clusters.



(a) t-SNE for GMVAE with no triplet loss and 15 clusters

(b) t-SNE for GMVAE with triplet loss and 15 clusters

Figure 4-5: Results of t-SNE on simulated product data

We compare this with the results of the baseline MDS. Again, the MDS was run with 4 different starting initialization, where the best one was used (the one that preserves distance in the embedding space the best).

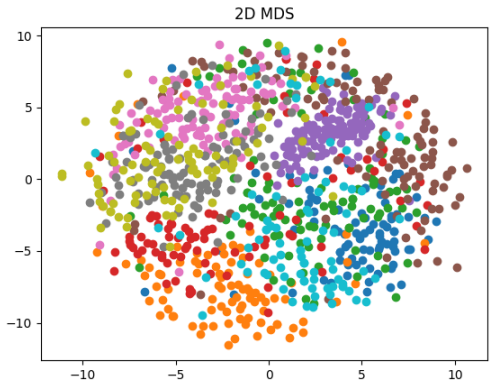


Figure 4-6: Results of MDS on simulated product data

The baseline MDS is qualitatively worse at separating the data clearly into clusters, but we do see meaningful clusters appear in this baseline as well.

4.2 Experiment 2: MNIST Data

We perform our second experiment on the MNIST data set, a well known dataset of handwritten digit images [6]. We describe in this subsection the data, the model

architecture, and the results where we compare the performance of the GMVAE with the MDS baseline.

4.2.1 Data

Number of Training Data	50000
Number of Test Data	10000
Number of "Unknown" Latent Classes	10
Data Dimensionality	Size 784 Vector
Data Type	Real-Valued

Table 4.3: Data summary of MNIST data

The data is the MNIST dataset, and consists of $\mathbf{x}$ vectors of size 784 (representing an image that is 28 x 28 pixels). Each value of the vector is a float between 0 and 1, where 0 represents an empty pixel and 1 represents a full pixel. The data is approximately distributed evenly (there are equal number of images for each digit class). The MNIST image data is used directly in the net without any preprocessing.

The triplet data is generated similarly to the process described in 4.1.1, in the following way: we choose an anchor randomly from the available data. We pick a "positive" from the same digit class as the anchor, and a "negative" randomly from the other classes.

4.2.2 Model Architecture

The architecture is the same as the architecture described in Section 4.1.2

4.2.3 Results

We vary the number of clusters and whether triplet loss is used or not when running the model on the data. The results are below.

We can visualize the effectiveness of the clustering in 2D by applying t-distributed Stochastic Neighbor Embedding (t-SNE) to the embedding created by the GMVAE.

Number of Clusters (k)	Triplet	Test Accuracy
10	No	0.557
	Yes	0.663
15	No	0.662
	Yes	0.768
20	No	0.700
	Yes	0.835

Table 4.4: Product latent category prediction accuracy of the GMVAE after 1600000 training points

The results are shown below; note that each different color represents a different digit class. We see that the GMVAE performs reasonably well, since the t-SNE does show some separation of the clusters.

We compare this with the results of the baseline MDS. This MDS was run with 4 different starting initialization, where the best one was used (the one that preserves distance in the embedding space the best). Again, each color represents a different digit class.

This MDS baseline is qualitatively worse on separating the different clusters from each other.

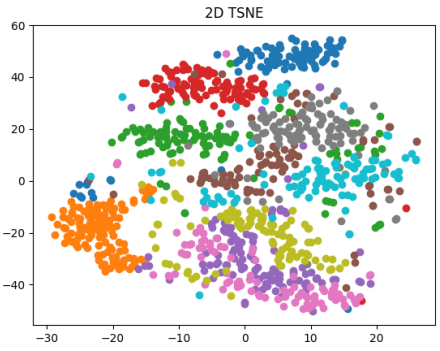
4.3 Experiment 3: Amazon Fashion Data

Our third experiment is on the Amazon Fashion dataset [16]. This is a dataset of product images, product categories, and similar products (products that are viewed together).

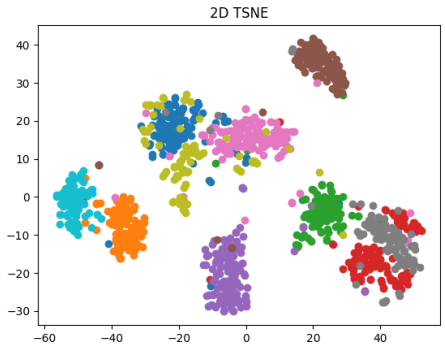
4.3.1 Data

Number of Training Data	72000
Number of Test Data	18000
Number of "Unknown" Latent Classes	6
Data Dimensionality	Size (128, 128, 3) Vector
Data Type	Integer between 0 and 255

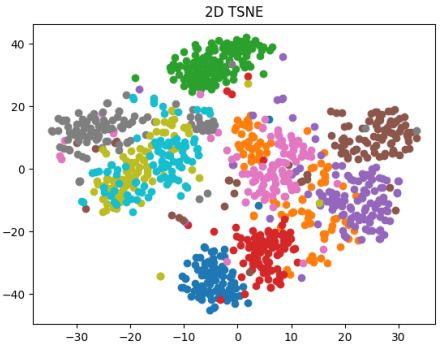
Table 4.5: Data summary of the Amazon Fashion dataset



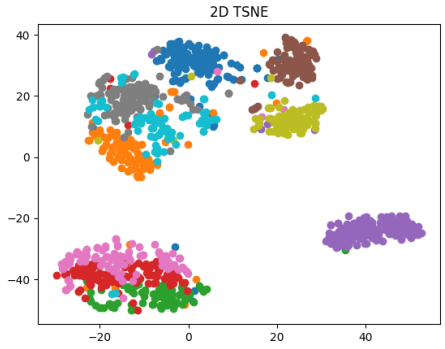
(a) t-SNE for GMVAE with no triplet loss and 10 clusters



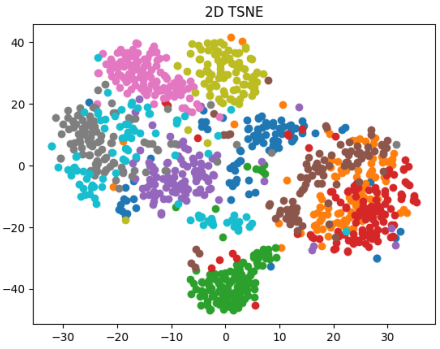
(b) t-SNE for GMVAE with triplet loss and 10 clusters



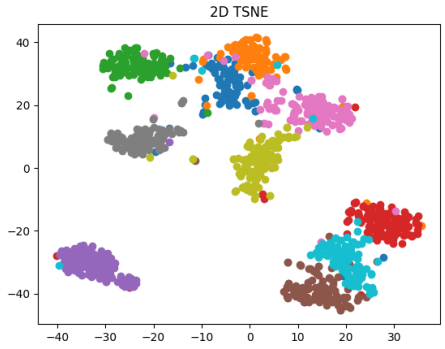
(c) t-SNE for GMVAE with no triplet loss and 15 clusters



(d) t-SNE for GMVAE with triplet loss and 15 clusters



(e) t-SNE for GMVAE with no triplet loss and 20 clusters



(f) t-SNE for GMVAE with triplet loss and 20 clusters

Figure 4-7: Results of t-SNE on MNIST data

The image data that we cluster are JPEG images of different sizes. The images are in RGB format, so the data for each image has shape $(a, b, 3)$, where a represents the height in pixels and b represents the width in pixels. Associated with each image

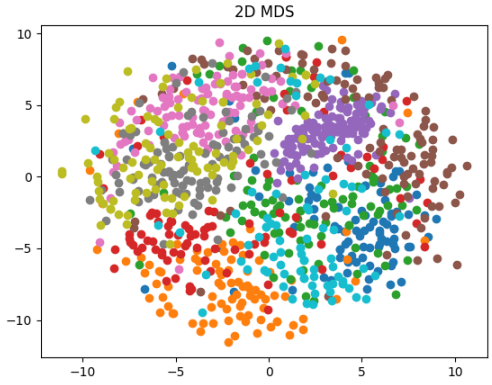


Figure 4-8: Results of MDS on MNIST data

is also a category, a one-hot vector of size 6 representing which of the six categories (mens pants, womens pants, men shirts, women shirts, men shoes, and female shoes) the image belongs to.

The products that are viewed together are used to generate the triplet data in the following way:

1. We let product $\mathbf{x}^a$ be the anchor.
2. We let each product that is viewed together with $\mathbf{x}^a$ by the "positive", $\mathbf{x}^p$.
3. We randomly sample a "negative", $\mathbf{x}^n$ from the rest of the dataset. It may be possible for this negative to be in the same category as the anchor, since we do not use labels for training.

We first preprocess the data by resizing all of the images to be 128 pixels by 128 pixels. Each value of the resulting matrix of size $(128, 128, 3)$ is an integer between 0 and 255. In order to get the values between 0 and 1, we also divide by 255. The triplet data is preprocessed in the same way.

4.3.2 Model Architecture

The model has four different component nets that has the same function as the nets described in experiment 1, but very different architecture. The new architecture for the image data is described below.

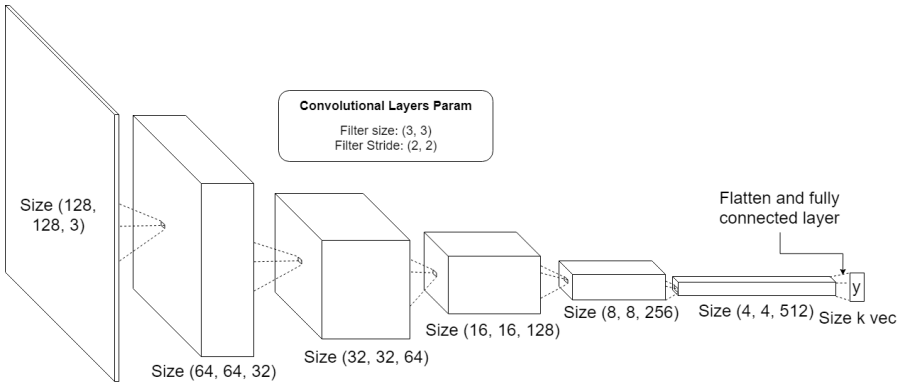


Figure 4-9: Architecture of the first component neural net on the Amazon Fashion dataset, the encoder for latent product categories

1. An encoder net that takes the image data, $\mathbf{x}$, and predicts a cluster $\mathbf{y}$. This net consists of five convolutional layers and one fully connected layer, where the five convolutional layers uses ReLU activation. Each convolutional layer has kernel size of $(3, 3)$, stride of size $(2, 2)$, and filter sizes 32, 64, 128, 256, and 512 respectively. We flatten the last convolutional layer, which has output of shape $(4, 4, 512)$, and feed the flattened vector, which has length 8192, to the dense layer. The last layer is of size k, where softmax activation gives the probability of the image being in each cluster.

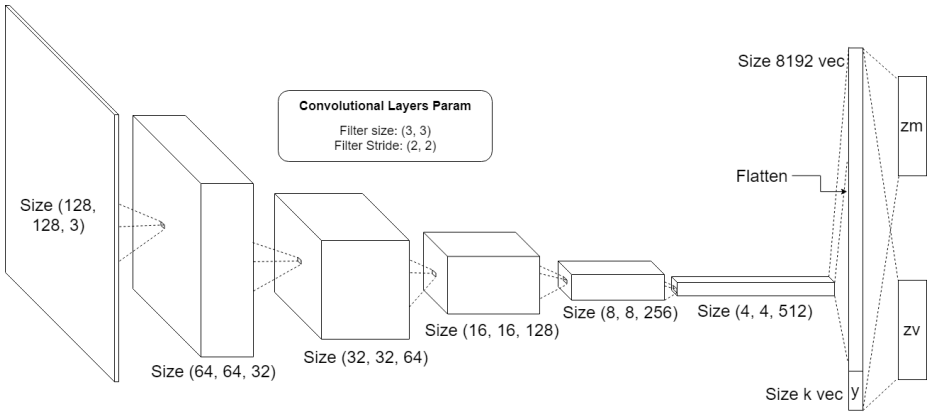


Figure 4-10: Architecture of the second component neural net on the Amazon Fashion dataset, the encoder for perceptual map embedding

2. An encoder net that take the image data, $\mathbf{x}$, and predicted cluster $\mathbf{y}$, to create the embedding $\mathbf{z}$ (with dimension reduction). This net takes as input the vector

$\mathbf{x}$ and runs it through 5 convolutional layers (the same layers as described in net 1). The result of the last convolutional layer, a vector of shape $(4, 4, 512)$, is flattened and concatenated to $\mathbf{y}$. This is fed to two dense layers: the first dense layer outputs the mean of the embedding ($\mathbf{z_m}$) and the second dense layer outputs the variance of the embedding ($\mathbf{z_v}$). The second dense layer uses softplus activation to ensure that variance is positive.

3. An encoder net that takes the predicted cluster, $\mathbf{y}$, and predicts the Gaussian prior for the cluster in the embedding space. This has the same architecture as net 3 in Section 4.1.2.

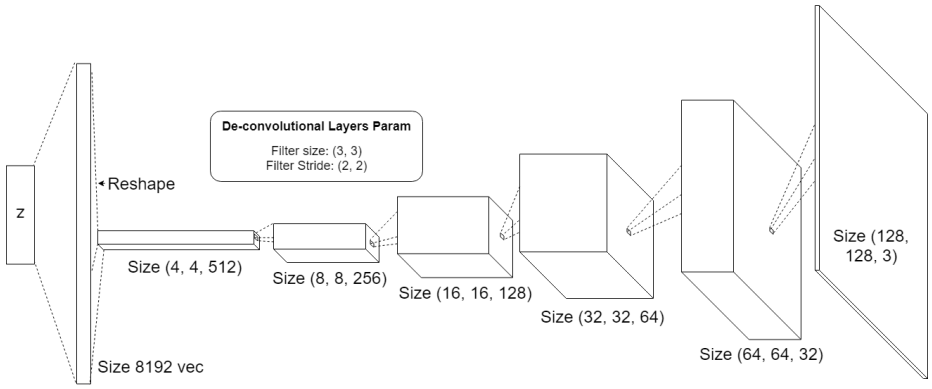


Figure 4-11: Architecture of the fourth component neural net on the Amazon Fashion dataset, the decoder net of the perceptual map embedding. Both the $\mathbf{x}_m$ and the $\mathbf{x}_v$ net have the same architecture.

4. A decoder net that takes the embedding, z, and tries to reconstruct the image x_r. It assumes a Gaussian distribution of x given z. The net outputs two different values, the mean of the predicted reconstruction, x_m, and the variance of the predicted reconstruction, x_v. There are 10 different deconv layers and 2 fully connected layers. The first fully connected layer is a dense layer with ReLU activation that outputs a vector of size 8192 and is resized into a vector of shape $(4, 4, 512)$. This result is fed into the 5 deconv layers that are opposite the layers described in net 1 (has kernel size $(3, 3)$, stride size $(2, 2)$, and filter sizes 256, 128, 64, 32, and 3 respectively). The output uses sigmoid activation, and gives the value of x_m. The second fully connected layer and deconvolutional

layers are exactly the same. We use sigmoid activation for the variance in this case, since we know that the variance of the data must be less than 1 (since all data points are between 0 and 1).

4.3.3 Results

We vary the number of clusters and whether triplet loss is used or not when running the model on the data. The results are below.

Number of Clusters (k)	Triplet	Test Accuracy
6	No	0.397
	Yes	0.441
12	No	0.362
	Yes	0.428
15	No	0.298
	Yes	0.25

Table 4.6: Product latent category prediction accuracy of the GMVAE after 134400 training points

We can visualize the effectiveness of the clustering in 2D by applying t-distributed Stochastic Neighbor Embedding (t-SNE) to the embedding created by the GMVAE. The results are shown below.

We compare this with the results of the baseline MDS. This MDS was run with 4 different starting initialization, where the best one was used (the one that preserves distance in the embedding space the best).

This MDS baseline is qualitatively worse on separating the different clusters from each other.

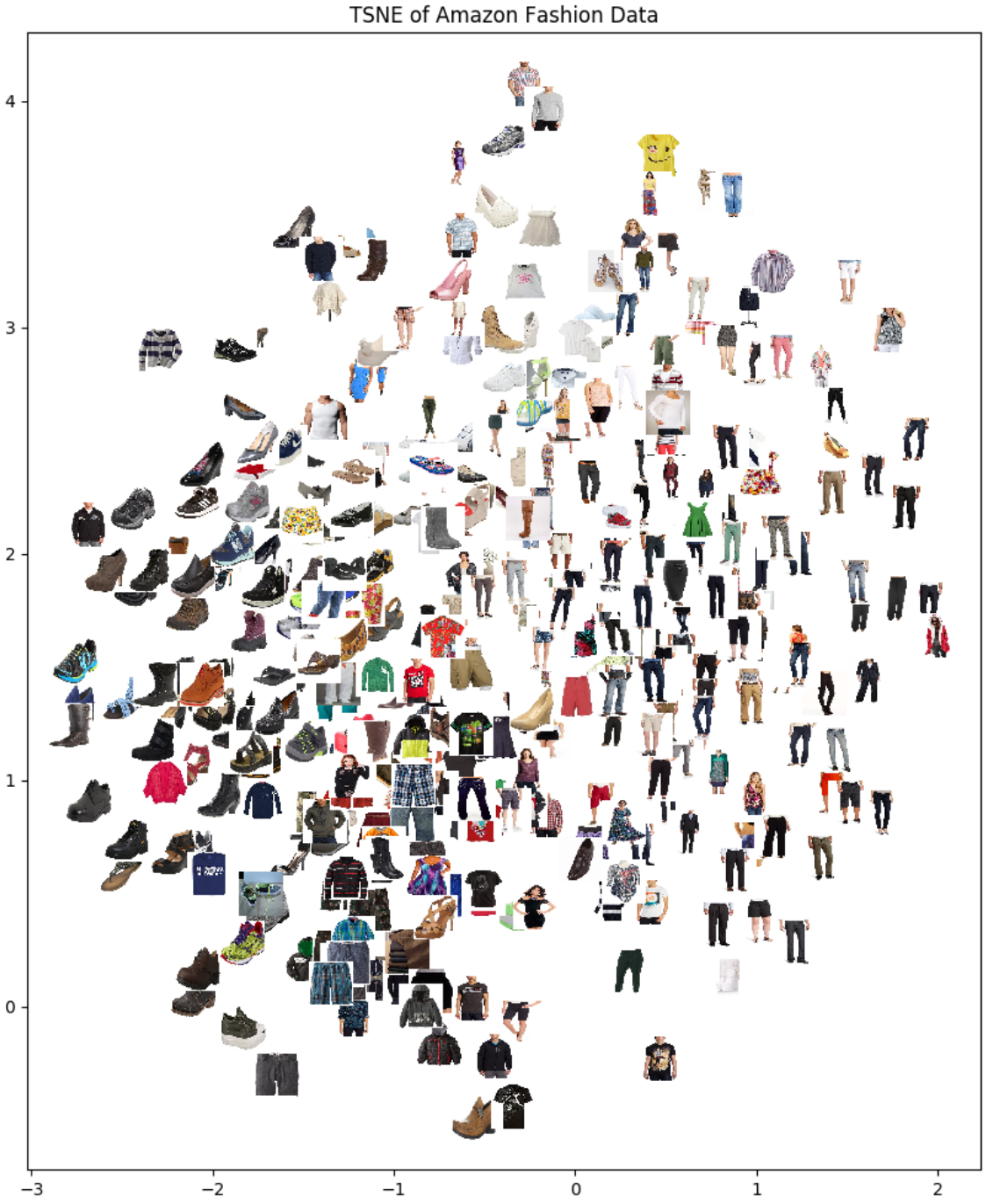


Figure 4-12: t-SNE on the Amazon Fashion dataset using 6 clusters and triplet loss.

Figure 4-13: Results of MDS on the Amazon Fashion dataset

Chapter 5

Discussions and Limitations

In this chapter, we observe and discuss how our GMVAE with triplet loss performs. Specifically, we will talk about the effectiveness of the model compared with MDS, the effect of the number of clusters on accuracy and computation time, the effectiveness of triplet loss on the model, and limitations of the model.

5.1 Comparison with MDS

We currently do not have a quantitative method with which to compare MDS and our GMVAE approach for perceptual mapping. Without running some clustering algorithm on the MDS result, we cannot actually assign labels to individual images. This is another advantage our approach has over baseline MDS and PCA methods.

The perceptual maps produced by running t-SNE on the embeddings have far clearer clusters for the simulated data and for MNIST (refer to Figures 4-5 and 4-6 for custom data, 4-7 and 4-8 for MNIST data). For Amazon Fashion data, our method performs comparable to but still better than the MDS baseline (refer to Figures 4-12 and 4-13).

For the Amazon Fashion dataset, our results are somewhat comparable to the baseline - this may by the difficulty in separating out similar products. For example, the real categories are male shirts, female shirts, male pants, female pants, male shoes, and females shoes. The model seems to have difficultly separating out the difference

between male and female versions of the same category.

5.2 Effect of Number of Training Clusters

This metric is relevant to both the MNIST dataset and the Amazon Fashion dataset. For the MNIST dataset, increasing the number of clusters increases speed of convergence (refer to Table 4.4). However, for the Amazon Fashion dataset, increasing the number of clusters decreases speed of convergence (refer to Table 4.6).

For MNIST, we know from the labels that there are 10 latent clusters. Therefore, we would expect that as long as we train with at least 10 different clusters in the latent space, then we should be able to achieve comparable accuracy no matter how many more clusters we use. However, for MNIST at least, training with more clusters seems to improve the convergence speed of the model. One possible reason using more clusters is advantageous is that it could help prevent local minima. With more clusters, it becomes less likely that all clusters get stuck in either local minima or areas of weaker gradients, which could decrease the speed of training.

For Amazon Fashion, we know from the labels that are 6 latent clusters. Unlike the MNIST dataset, it seems that increasing the number of latent clusters actually decreases the speed of convergence. This may be due to the complexity of the data compared to MNIST - with more clusters, perhaps it becomes more likely that some of the clusters are based on significant features that might span several actual groups (such as a color or logo).

Lastly, we note that increasing the number of clusters the model trains with significantly increases training time with the same number of epochs. This is because, as mentioned in Section 3.2, the number of operations of triplet loss scales cubicly with the number of clusters used by the model. Fortunately, memory use scales linearly, so it is still possible to train large neural nets with many clusters albeit very slowly.

5.3 Effectiveness of Triplet Loss

From the results, it seems that triplet loss is effective, as expected. By increasing the amount of data available to the model through triplet loss, we would expect the model to perform better. We describe in this section however the two advantages of using triplet loss, and how these advantages materialize through the loss function of triplet loss.

We write the loss function of triplet loss, Equation 3.1 here for reference:

$$\mathbb{E}_{q_\phi(\mathbf{z}^{(a)},\mathbf{y}^{(a)},\mathbf{z}^{(p)},\mathbf{y}^{(p)},\mathbf{z}^{(n)},\mathbf{y}^{(n)}|\mathbf{a},\mathbf{p},\mathbf{n}}\max(0,\alpha+E(\mathbf{a})\cdot E(\mathbf{n})-E(\mathbf{a})\cdot E(\mathbf{p}) \tag{5.1}$$

where E is the encoder as shown in Figure 3-2 a. The α term in the expression is the margin, a hyperparameter that we have set to 0.2 in our experiments.

The first advantage of using triplet loss is the increase in cluster accuracy (refer to Tables 4.2, 4.4, and 4.6). The increase in accuracy corresponds to decreasing the loss function above by changing the probability function $q_\phi(\mathbf{z^{(a)}},\mathbf{y^{(a)}},\mathbf{z^{(p)}},\mathbf{y^{(p)}},\mathbf{z^{(n)}},\mathbf{y^{(n)}}|\mathbf{a},\mathbf{p},\mathbf{n})$ so that similar products $\mathbf{a}$ and $\mathbf{p}$ would have a higher probability of $p(\mathbf{y^a}=\mathbf{y^p})$ and dissimilar products $\mathbf{a}$ and $\mathbf{n}$ would have a lower probability of $p(\mathbf{y^a}=\mathbf{y^n})$.

The second advantage of using triplet loss is creating tighter clusters, where the embeddings of similar products are closer together (this is apparent in Figure 4-7). This change corresponds to the second way of decreasing the above loss function, which is to directly decrease the value $\alpha+E(\mathbf{a})\cdot E(\mathbf{n})-E(\mathbf{a})\cdot E(\mathbf{p})$. Decreasing this value allows the neural net to learn to make similar products have more similar embeddings, creating a tighter perceptual map.

5.4 Limitations

In this section, we will note the major limitations of the our model.

The first limitation is the model only operates on data in the format specified in Section 3.1. In order for the model to be effective, we need both unlabeled product

images and triplet data. However, it may be nontrivial to gather triplet data for products. One application where triplet data is readily available though is recommender systems, which keeps track of products that are often viewed together. This data could be used to generate triplets, similar to how we generated triplet from similar products in the Amazon Fashion dataset (Section 4.3).

A second limitation is that the model is not effective on data with many different latent product categories. As describe in Section 3.2, triplet loss scales cubicly with the number of latent clusters. Therefore, if there are too many clusters, using triplet loss becomes prohibitively expensive. One workaround to this problem is to try to run the model hierarchically on data with many different latent clusters. For example, on a dataset with 25 different latent clusters, we can run the model assuming 5 different latent clusters. Then, for each of the 5 different latent clusters, we can run the model again assuming 5 different latent sub-clusters, thereby separating the latent space into 25 latent clusters. However, this approach does not utilize information on the relationship between sub-clusters, and is not optimal.

A third limitation is that the model may require skillful balancing of hyperparameters in order to produce non-degenerate clusters. Early in training, kl loss (Section 3.1) could dominate and produce large gradients in the net. This could cause the model to oversimplify, such as creating a simple Gaussian in the embedding space rather than a Gaussian mixture, which would decrease the KL loss. Some regularization terms, such a negative entropy on the encoder for latent product categories, may be needed to decrease anti-clustering tendencies.

Chapter 6

Conclusion

Understanding how customers perceive products is a fundamental task in marketing and product design, including: identifying gaps in the market, understanding competition, and finding how new products fit into a market. However, qualitative approaches to perceptual mapping such as surveys are difficult to design and labor intensive, and popular quantitative approaches to perceptual mapping (such as principal component analysis (PCA), multidimensional scaling (MDS), and autoencoders) have several key disadvantages. For example, PCA is only capable of linear transformations, MDS needs a set distance function so is difficult to include information such as triplets, and autoencoders lead to embeddings where it becomes unfeasible to infer similarity.

We propose and evaluate a more recent technique to the problem where we have images or products and lists of products that are related to each other. By using a Gaussian mixture variational autoencoder (GMVAE) with triplet loss, we are able to produce perceptual maps that are qualitatively better than the MDS baseline across three experiments, including: experiments with simulated data (simple data with pre-made clusters), MNIST data (famous dataset of handwritten digits), and the Amazon Fashion dataset (dataset of product images, product categories, and related products). Also, We find quantitatively that we are able to achieve high accuracies with unsupervised clustering. Triplet loss is important, as its use has increase accuracy across every single experiment as well.

Appendix A

Nomenclature

Model Nomenclature

1. Feature Data:
   (a) $\mathbf{x}^i$: The i^{th} input image available to the model
   (b) $\mathbf{y}$: The latent product category of an input vector $\mathbf{x}$. This is not known during training, and is used to evaluate the model.
   (c) k : The total number of latent product categories that is assumed by the model.
   (d) $\mathbf{y}_{k_i}$ The latent product category that vector $\mathbf{x}^i$ belongs to.
2. Triplet Data, of form $(\mathbf{a}, \mathbf{p}, \mathbf{n})$:
   (a) $\mathbf{a}$: The anchor vector of the triplet. This is the vector with which similarity is measured with and compared to.
   (b) $\mathbf{p}$: The positive vector of the triplet. The positive vector is closely related to the anchor vector.
   (c) $\mathbf{n}$: The negative vector of the triplet. This negative vector is not related to the anchor vector.
3. Model Outputs
   (a) $\mathbf{z}^i$: The model embedding for input vector $\mathbf{x}^i$.
   (b) $\mathbf{x}_r^i$: The reconstructed input when the model is given embedding $\mathbf{z}^i$
   (c) E : The encoder such that $\mathbf{z}^i = E(\mathbf{x}^i)$
   (d) D : The decoder such that $\mathbf{x}_r^i = D(\mathbf{z}^i)$

Figure A-1: Summary of Nomenclature

Bibliography

[1] David A Aaker, Vineet Kumar, and George S Day. *Marketing research*. John Wiley & Sons, 2008.

[2] Klaus-Jürgen Bathe and Edward L Wilson. Numerical methods in finite element analysis. 1976.

[3] Leann Lipps Birch. Dimensions of preschool children's food preferences. *Journal of nutrition education*, 11(2):77–80, 1979.

[4] Trevor F Cox and Michael AA Cox. *Multidimensional scaling*. Chapman and hall/CRC, 2000.

[5] Mark L Davison. Multidimensional scaling. 1991.

[6] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[7] Wayne S DeSarbo and Donna L Hoffman. Constructing mds joint spaces from binary choice data: A multidimensional unfolding threshold model for marketing research. *Journal of Marketing Research*, pages 40–54, 1987.

[8] Jeffrey P Dotson, Mark A Beltramo, Elea McDonnell Feit, and Randall C Smith. Modeling the effect of images on product choices. 2016.

[9] Rob A Dunne and Norm A Campbell. On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function. In *Proc. 8th Aust. Conf. on the Neural Networks, Melbourne*, volume 181, page 185. Citeseer, 1997.

[10] Pauline Faye, Damien Brémaud, Eric Teillet, Philippe Courcoux, Agnès Giboreau, and Huguette Nicod. An alternative to external preference mapping based on consumer perceptive mapping. *Food quality and preference*, 17(7-8):604–614, 2006.

[11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

[12] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[13] Paul E Green. *Multidimensional scaling and related techniques in marketing analysis*. Allyn and Bacon, 1970.

[14] Joseph F Hair, Mary Celsi, David J Ortinau, and Robert P Bush. *Essentials of marketing research*. McGraw-Hill/Higher Education New York, NY, 2008.

[15] John R Hauser and Frank S Koppelman. Alternative perceptual mapping techniques: Relative accuracy and usefulness. *Journal of marketing Research*, pages 495–506, 1979.

[16] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517. International World Wide Web Conferences Steering Committee, 2016.

[17] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017.

[18] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[19] Ian Jolliffe. Principal component analysis. In *International encyclopedia of statistical science*, pages 1094–1096. Springer, 2011.

[20] Namwoo Kang, Yi Ren, Fred M Feinberg, and Panos Y Papalambros. Form+ function: Optimizing aesthetic product design via adaptive, geometrized preference elicitation. 2016.

[21] Dong Jin Kim, Woo Gon Kim, and Jin Soo Han. A perceptual mapping of online travel agencies and preference attributes. *Tourism management*, 28(2):591–603, 2007.

[22] D. P Kingma and M. Welling. Auto-Encoding Variational Bayes. *ArXiv e-prints*, December 2013.

[23] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[24] Yan Liu, Krista J Li, Haipeng Chen, and Subramanian Balachander. The effects of productsâĂŹ aesthetic design on demand and marketing-mix effectiveness: The role of segment prototypicality and brand consistency. *Journal of Marketing*, 81(1):83–102, 2017.

[25] HJH MacFie and DMH Thomson. Preference mapping and multidimensional scaling. *Sensory analysis of foods/edited by JR Piggott*, 1988.

[26] NK Malhotra, DF Birks, A Palmer, and N Koenig-Lewis. Market research: an applied approach. *Journal of marketing management*, 27:1208–1213, 2003.

[27] Lívia Markíczy and Jeff Goldberg. A method for eliciting and comparing causal maps. *Journal of management*, 21(2):305–333, 1995.

[28] K Sridhar Moorthy and Ivan PL Png. Market segmentation, cannibalization, and the timing of product introductions. *Management Science*, 38(3):345–359, 1992.

[29] Nasser M Nasrabadi. Pattern recognition and machine learning. *Journal of electronic imaging*, 16(4):049901, 2007.

[30] Michael A Nestrud and Harry T Lawless. Perceptual mapping of citrus juices using projective mapping and profiling data from culinary professionals and consumers. *Food quality and preference*, 19(4):431–438, 2008.

[31] Pascale G Quester, Robyn L McGuiggan, William D Perreault, and E Jerome McCarthy. *Marketing: creating and delivering value*. 2004.

[32] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, page 4. ACM, 2014.

[33] Wei Wang, Yan Huang, Yizhou Wang, and Liang Wang. Generalized autoencoder: A neural network framework for dimensionality reduction. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 490–497, 2014.